# Galactic Token Sliding

**Valentin Bartier**
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

**Nicolas Bousquet**
Univ Lyon, CNRS, UCBL, INSA Lyon, LIRIS, UMR5205, France

**Amer E. Mouawad**
American University of Beirut, Lebanon
University of Bremen, Germany

## Abstract

Given a graph $G$ and two independent sets $I_s$ and $I_t$ of size $k$, the Independent Set Reconfiguration problem asks whether there exists a sequence of independent sets (each of size $k$) $I_s = I_0, I_1, I_2, \ldots, I_\ell = I_t$ such that each independent set is obtained from the previous one using a so-called reconfiguration step. Viewing each independent set as a collection of $k$ tokens placed on the vertices of a graph $G$, the two most studied reconfiguration steps are token jumping and token sliding. In the Token Jumping variant of the problem, a single step allows a token to jump from one vertex to any other vertex in the graph. In the Token Sliding variant, a token is only allowed to slide from a vertex to one of its neighbors. Like the Independent Set problem, both of the aforementioned problems are known to be W[1]-hard on general graphs (for parameter $k$). A very fruitful line of research [5, 14, 27, 25] has showed that the Independent Set problem becomes fixed-parameter tractable when restricted to sparse graph classes, such as planar, bounded treewidth, nowhere-dense, and all the way to biclique-free graphs. Over a series of papers, the same was shown to hold for the Token Jumping problem [17, 22, 26, 8]. As for the Token Sliding problem, which is mentioned in most of these papers, almost nothing is known beyond the fact that the problem is polynomial-time solvable on trees [11] and interval graphs [6]. We remedy this situation by introducing a new model for the reconfiguration of independent sets, which we call galactic reconfiguration. Using this new model, we show that (standard) Token Sliding is fixed-parameter tractable on graphs of bounded degree, planar graphs, and chordal graphs of bounded clique number. We believe that the galactic reconfiguration model is of independent interest and could potentially help in resolving the remaining open questions concerning the (parameterized) complexity of Token Sliding.

**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability; Theory of computation → W hierarchy

**Keywords and phrases** reconfiguration, independent set, galactic reconfiguration, sparse graphs, token sliding, parameterized complexity

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2022.15

**Related Version** *Full Version*: https://arxiv.org/abs/2204.05549

**Funding** This work is supported by PHC Cedre project 2022 "PLR".
*Valentin Bartier*: Supported by ANR project GrR (ANR-18-CE40-0032).
*Nicolas Bousquet*: Supported by ANR project GrR (ANR-18-CE40-0032).
*Amer E. Mouawad*: Research supported by the Alexander von Humboldt Foundation and partially supported by URB project "A theory of change through the lens of reconfiguration".

## 1 Introduction

Many algorithmic questions can be posed as follows: given the description of a system state and the description of a state we would "prefer" the system to be in, is it possible to transform the system from its current state into the more desired one without "breaking" the system in the process? And if yes, how many steps are needed? Such problems


naturally arise in the fields of mathematical puzzles, operational research, computational geometry [23], bioinformatics, and quantum computing [13] for instance. These questions received a substantial amount of attention under the so-called *combinatorial reconfiguration framework* in the last few years [9, 28, 30]. We refer the reader to the surveys by van den Heuvel [28] and Nishimura [24] for more background on combinatorial reconfiguration.

In this work, we focus on the reconfiguration of independent sets. Given a simple undirected graph $G$, a set of vertices $S \subseteq V(G)$ is an *independent set* if the vertices of $S$ are all pairwise non-adjacent. Finding an independent set of maximum cardinality, i.e., the Independent Set problem, is a fundamental problem in algorithmic graph theory and is known to be not only NP-hard, but also W[1]-hard and not approximable within $\mathcal{O}(n^{1-\epsilon})$, for any $\epsilon > 0$, unless P = NP [31].

We view an independent set as a collection of tokens placed on the vertices of a graph such that no two tokens are adjacent. This gives rise to two natural adjacency relations between independent sets (or token configurations), also called *reconfiguration steps*. These two reconfiguration steps, in turn, give rise to two combinatorial reconfiguration problems. In the Token Jumping (TJ) problem, introduced by Kamiński et al. [20], a single reconfiguration step consists of first removing a token on some vertex $u$ and then immediately adding it back on any other vertex $v$, as long as no two tokens become adjacent. The token is said to *jump* from vertex $u$ to vertex $v$. In the Token Sliding (TS) problem, introduced by Hearn and Demaine [15], two independent sets are adjacent if one can be obtained from the other by a token jump from vertex $u$ to vertex $v$ with the additional requirement of $uv$ being an edge of the graph. The token is then said to *slide* from vertex $u$ to vertex $v$ along the edge $uv$. Note that, in both the TJ and TS problems, the size of independent sets is fixed. Generally speaking, in the Token Jumping and Token Sliding problems, we are given a graph $G$ and two independent sets $I_s$ and $I_t$ of $G$. The goal is to determine whether there exists a sequence of reconfiguration steps – a *reconfiguration sequence* – that transforms $I_s$ into $I_t$ (where the reconfiguration step depends on the problem).

Both problems have been extensively studied, albeit under different names [6, 7, 11, 12, 16, 19, 20, 22]. It is known that both problems are PSPACE-complete, even on restricted graph classes such as graphs of bounded bandwidth (and hence pathwidth) [29] and planar graphs [15]. On the positive side, it is easy to prove that Token Jumping can be decided in polynomial time on trees (and even on chordal graphs) since we simply have to iteratively jump tokens to leaves (resp. vertices that only appear in the bag of a leaf in the clique tree) to transform an independent set into another. Unfortunately, for Token Sliding, the problem becomes more complicated because of what we call the *bottleneck effect*. Indeed, there might be a lot of empty leaves in the tree but there might be a bottleneck in the graph that prevents us from reaching these desirable vertices. For instance, consider a star plus a long subdivided path attached to the center of the star. One cannot move any token from the leaves of the star to the path if there are at least two tokens on the leaves (in other words, two tokens adjacent to a cut-vertex prevent us from using the cut vertex). Even if we can overcome this issue for instance on trees [11] and on interval graphs [6], the Token Sliding problem remains much "harder" than the Token Jumping problem. In split graphs for instance (which are chordal), Token Sliding is PSPACE-complete [4]. Lokshtanov and Mouawad [21] showed that, in bipartite graphs, Token Jumping is NP-complete while Token Sliding remains PSPACE-complete.

In this paper we focus on the parameterized complexity of the Token Sliding problem. While the complexity of Token Jumping parameterized by the size of the independent set is quite well understood, the comprehension of the complexity of Token Sliding remains

evasive. A problem $\Pi$ is FPT (fixed-parameterized tractable) parameterized by $k$ if one can solve it in time $f(k) \cdot poly(n)$, for some computable function $f$. In other words, the combinatorial explosion can be restricted to a parameter $k$. In the rest of the paper, our parameter $k$ will be the size of the independent set (i.e. number of tokens). Both Token Jumping and Token Sliding are known to be W[1]-hard[1] parameterized by $k$ on general graphs [22]. On the positive side, Lokshtanov et al. [22] showed that Token Jumping is FPT on bounded degree graphs. This result has been extended in a series of papers to planar graphs, nowhere-dense graphs, and finally strongly $K_{\ell,\ell}$-free graphs [18, 8], a graph being strongly $K_{\ell,\ell}$-free if it does not contain any $K_{\ell,\ell}$ as a subgraph. For Token Sliding, it was proven in [2] that the problem is W[1]-hard on bipartite graphs and $C_4$-free graphs (a similar result holds for Token Jumping but based on weaker assumptions for the bipartite case [1]).

Almost no positive result is known for Token Sliding even for incredibly simple cases like graphs of bounded degree. Our main contributions are to develop two general tools for the design of parameterized algorithms for Token Sliding, namely galactic reconfiguration and types. Galactic reconfiguration is a general simple tool that allows us to reduce instances. Using it, we will derive that Token Sliding is FPT on bounded degree graphs. Our second tool, called types, will in particular permit to show that when the deletion of a small subset of vertices leaves too many components, then one of them can be removed. Combining both tools with additional rules, we prove that Token Sliding is FPT on planar graphs and on chordal graphs of bounded clique number. We complement these results by proving that Token Sliding is W[1]-hard on split graphs.

Our first result is the following:

▶ **Theorem 1.1.** Token Sliding *is FPT on bounded degree graphs parameterized by* $k$.

Much more than the result itself, we believe that our main contribution here is the general framework we developed for its proof, namely galactic reconfiguration. Before explaining exactly what it is, let us explain the intuition behind it. As we already said, even if there are vertices which are far apart from the vertices of an independent set, we are not sure we can reach them because of the bottleneck effect. Our intuition was that it should be possible to reduce a part of large diameter of the graph that does not contain any tokens (just as we can find irrelevant vertices when we have large grid minors). The idea is that since the diameter is large, we should be able to hide tokens far apart from each other in this structure, avoiding the "bottleneck issue". And thus the structure should be reducible. However, proving that a structure can be reduced in reconfiguration is usually very technical. To overcome this problem, we introduce a new kind of vertices called *black holes* which can swallow as many tokens of the independent set as we like. A *galactic graph* is a graph that might contain black holes. A *galactic independent set* is a set of vertices on which tokens lie, such that the set of non black hole vertices holding tokens is an independent set and such that each black hole might contain any number of tokens.

Our main result is to prove that if there exists a long shortest path that is at distance at least two from the initial and target independent sets, then we can replace it by a black hole (whose neighborhood is the union of the neighborhoods of the path vertices). This rule, together with other simple rules on galactic graphs, allows us to reduce the size of bounded-degree graphs until they reach a size of at most $f(k)$ in polynomial time. Since a kernel ensures the existence of an FPT algorithm, Theorem 1.1 holds.

[1] Informally, it means that they are very unlikely to admit an FPT algorithm.

In the rest of the paper, we combine galactic graphs with other techniques to prove that Token Sliding is FPT on several other graph classes. We first prove the following:

▶ **Theorem 1.2.** Token Sliding *is FPT on planar graphs parameterized by $k$.*

To prove Theorem 1.2, we cannot simply use our previous long path reduction since, in a planar graph, there might be a universal vertex which prevents the existence of a long shortest path. Note that the complexity of Token Sliding is open on outerplanar graphs and it was not known to be FPT prior to our work.

Our strategy consists in reducing to planar graphs of bounded degree and then applying Theorem 1.1. To do so, we provide another general tool to reduce graphs for Token Sliding. Namely, we show that if there is a set $X$ of vertices such that $G - X$ contains too many connected components (in terms of $k$ and $|X|$) then at least one of them can be safely removed. The idea of the proof consists in defining the *type* of a connected component of $G - X$. From a very high level perspective, the type[2] of a path in a component of $G - X$ is the sequence of its neighborhoods in $X$. The type of a component $C$ is the union of the types of the paths starting on a vertex of $C$. We then show that if too many components of $G - X$ have the same type then one of them can be removed.

However, this component reduction is not enough since, in the case of a vertex $x$ universal to an outerplanar graph, the deletion of $x$ does not leave many connected components. We prove that, we can also reduce a planar graph if (i) there are too many vertex-disjoint $(x, y)$-paths for some pair $x, y$ of vertices or, (ii) if a vertex has too many neighbors on an induced path. Since one can prove that in an arbitrarily large planar graph with no long shortest path (i) or (ii) holds, it will imply Theorem 1.2.

Note that our proof techniques can be easily adapted to prove that the problem is FPT for any graph of bounded genus. We think that the notion of types may be crucial to derive FPT algorithms on larger classes of graphs such as bounded treewidth graphs.

We finally provide another application of our method by proving that the following holds:

▶ **Theorem 1.3.** Token Sliding *is FPT on chordal graphs of bounded clique number.*

The proof of Theorem 1.3 consists in proving that, since there is a long path in the clique tree, we can either find a long shortest path (and we can reduce the graph using galactic rules) or find a vertex $x$ in a large fraction of the bags of this path. In the second case, we show that we can again reduce the graph. We complement this result by proving that it cannot be extended to split graphs, contrarily to Token Jumping.

▶ **Theorem 1.4.** Token Sliding *is W[1]-hard on split graphs.*

We show hardness via a reduction from the Multicolored Independent Set problem, known to be W[1]-hard [10]. The crux of the reduction relies on the fact that we have a clique of unbounded size and hence we can use different subsets of the clique to encode vertex selection gadgets and non-edge selection gadgets.

The first natural generalization of our result on chordal graphs of bounded clique size would be the following:

▶ **Question 1.1.** *Is* Token Sliding *FPT on bounded treewidth graphs? Or simpler, on bounded pathwidth graphs?*

[2] The exact definition is actually more complicated.

We did not succeed in answering Question 1.1 but we think that the method we used for Theorem 1.3 is a good starting point (with a much more involved analysis). Recall that the problem is PSPACE-complete on graphs of constant bandwidth for a large enough constant that is not explicit in the proof [29]. Note that our galactic reconfiguration rules directly ensure that Token Sliding is FPT on bounded bandwidth graphs and the multi-component reduction ensures that the problem is FPT for graphs of bounded treedepth. But even for bounded pathwidth, the situation is unclear. There are good indications to think that solving the bounded pathwidth case is the hardest step to obtain an FPT algorithm for bounded treewidth graphs. On the positive side, we simply know that the problem is polynomial time solvable on graphs of treewidth one (namely forests) [11] and the problem is open for outerplanar graphs, which are graphs of treewidth 2:

▶ **Question 1.2.** *Is* Token Sliding *polynomial-time solvable on outerplanar graphs? Triangulated outerplanar graphs?*

**Organization of the paper.** In Section 2, we formally introduce galactic graphs and provide our main reduction rules concerning such graphs, including the long shortest path reduction lemma. In Section 3, we introduce the notion of types and journeys and prove that if there are too many connected components in $G - X$ then at least one of them can be removed. In Section 4, we briefly describe our results for Token Sliding on planar graphs and chordal graphs of bounded clique number. Our hardness reduction for split graphs and all omitted proofs can be found in the full version of the paper [3].

## 2 Galactic graphs and galactic token sliding

We say that a graph $G = (V, E)$ is a *galactic graph* when $V(G)$ is partitioned into two sets $A(G)$ and $B(G)$ where the set $A(G) \subseteq V(G)$ is the set of vertices that we call *planets* and the set $B(G) \subseteq V(G)$ is the set of vertices that we call *black holes*. For a given graph $G'$, we write $G' \prec G$ whenever $|A(G')| < |A(G)|$ or, in case of equality, $|B(G')| < |B(G)|$. In the standard Token Sliding problem, tokens are restricted to sliding along edges of a graph as long as the resulting sets remain independent. This implies that no vertex can hold more than one token and no two tokens can ever become adjacent. In a galactic graph, the rules of the game are slightly modified. When a token reaches a black hole (a special kind of vertex), the token is *absorbed* by the black hole. This implies that a black hole can hold more than one token, in fact it can hold all $k$ tokens. Moreover, we allow tokens to be adjacent as long as one of the two vertices is a black hole (since black holes are assumed to make tokens "disappear"). On the other hand, a black hole can also "project" any of the tokens it previously absorbed onto any vertex in its neighborhood (be it a planet or a black hole). Of course, all such moves require that we remain an independent set in the galactic sense. We say that a set $I$ is a *galactic independent set* of a galactic graph $G$ whenever $G[I \cap A]$ is edgeless. To fully specify a galactic independent set $I$ of size $k$ containing more than one token on black holes, we use a weight function $\omega_I : V(G) \to \{0, \ldots, k\}$. Hence, $\omega_I(v) \leq 1$ whenever $v \in A(G)$, $\omega_I(v) \in \{0, \ldots, k\}$ whenever $v \in B(G)$, and $\sum_{v \in V(G)} \omega_I(v) = k$.

We are now ready to define the Galactic Token Sliding problem. We are given a galactic graph $G$, an integer $k$, and two galactic independent sets $I_s$ and $I_t$ such that $|I_s| = |I_t| = k \geq 2$ (when $k = 1$ the problem is trivial). The goal is to determine whether there exists a sequence of token slides that will transform $I_s$ into $I_t$ such that each intermediate set remains a galactic independent set. As for the classical Token Sliding problem, given a galactic graph $G$ we can define a reconfiguration graph which we call the *galactic*

*reconfiguration graph* of $G$. It is the graph whose vertex set is the set of all galactic independent sets of $G$, two vertices being adjacent if their corresponding galactic independent sets differ by exactly one token slide. We always assume the input graph $G$ to be a connected graph, since we can deal with each component independently otherwise. Furthermore, components without tokens can be safely deleted. Given an instance $(G, k, I_s, I_t)$ of Galactic Token Sliding, we say that $(G, k, I_s, I_t)$ can be *reduced* if we can find an instance $(G', k', I'_s, I'_t)$ which is positive (yes-instance) if and only if $(G, k, I_s, I_t)$ is positive and $G' \prec G$.

Let $G$ be a galactic graph. A *planetary component* is a maximal connected component of $G[A]$. A *planetary path* $P$, or $A$-*path*, composed only of vertices of $A$, is called $A$-*geodesic* if, for every $x, y$ in $P$, $\mathsf{dist}_{G[A]}(x, y) = \mathsf{dist}_P(x, y)$. We use the term $A$-*distance* to denote the length of a shortest path between vertices $u, v \in A$ such that all vertices of the path are also in $A$. Let us state a few reduction rules that allow us to safely reduce an instance $(G, k, I_s, I_t)$ of Galactic Token Sliding to an instance $(G', k', I'_s, I'_t)$.

- Reduction rule R1 (adjacent black holes rule): If two black holes $u$ and $v$ are adjacent, we contract them into a single black hole $w$. If there are tokens on $u$ or $v$, the merged black hole receives the union of all such tokens. In other words, $\omega_{I'_s}(w) = \omega_{I_s}(u) + \omega_{I_s}(v)$ and $\omega_{I'_t}(w) = \omega_{I_t}(u) + \omega_{I_t}(v)$. Loops and multi-edges are ignored.
- Reduction rule R2 (dominated black hole rule): If there exists two black holes $u$ and $v$ such that $N(u) \subseteq N(v)$, $\omega_{I_s}(u) = 0$, and $\omega_{I_t}(u) = 0$, we delete $u$.
- Reduction rule R3 (absorption rule): If there exists $u, v$ such that $u$ is a black hole, $v \in N(u) \cap A$ ($v$ is a neighboring planet that could be in $I_s \cup I_t$) and $|((I_s \cup I_t) \cap A) \cap N[v]| \leq 1$, then we contract the edge $uv$. We say that $v$ is *absorbed* by $u$. If $v \in I_s \cup I_t$ then we update the weights of $u$ accordingly.
- Reduction rule R4 (twin planets rule): Let $u, v \in A(G)$ be two planet vertices that are *twins* (true or false twins). That is, either $uv \notin E(G)$ and $N(u) = N(v)$ or $uv \in E(G)$ and $N[u] = N[v]$. If $u \notin I_s \cup I_t$ then delete $u$. If both $u$ and $v$ are in $I_s$ (resp. $I_t$) and at least one of them is not in $I_t$ (resp. $I_s$) then return a trivial no-instance. If both $u$ and $v$ are in $I_s$ as well as $I_t$ then delete $N[u] \cup N[v]$, decrease $k$ by two, and set $I'_s = I_s \setminus \{u, v\}$ and $I'_t = I_t \setminus \{u, v\}$.
- Reduction rule R5 (path reduction rule): Let $G$ be a galactic graph and $P$ be a $A$-geodesic path of length at least $5k$ such that $(A \cap N[P]) \cap (I_s \cup I_t) = \emptyset$. Then, $P$ can be contracted into a black hole (we ignore loops and multi-edges). That is, we contract all edges in $P$ until one vertex remains.

▶ **Lemma 2.1.** *Reduction rule R5, the path reduction rule, is safe.*

**Sketch of the proof.** Let $P$ be an $A$-geodesic path of length $5k$ in $G$ such that no vertex of $A \cap N[P]$ are in the initial or target independent sets, $I_s$ and $I_t$. Let $G'$ be the graph obtained after contracting $P$ into a single black hole $b$. Let $I'_s$ and $I'_t$ be the galactic independent sets corresponding to $I_s$ and $I_t$. If there is a transformation from $I_s$ to $I_t$ in $G$, then one can show that there is a transformation from $I'_s$ to $I'_t$ in $G'$ by simply absorbing tokens that become adjacent to the black hole and then projecting them appropriately when needed.

Now we consider a transformation from $I'_s$ to $I'_t$ in $G'$ and show how to adapt it in $G$. We first prove (in the full version of the paper [3]) that we can always assume the existence of a sequence in $G'$ where the number of tokens in $N(b) \cap A$ is at most one throughout the sequence, for any black hole $b$. Now, assuming such a sequence, we can simulate the sequence in $G$ with the hard case being when multiple tokens slide into $b$. Note, however, that $P$ is of length $5k$ and is $A$-geodesic. Hence, every vertex $a \in A$ has at most three neighbors in $P$ and any independent set of size at most $k$ in $A$ has at most $3k$ neighbors in $P$. This leaves

$2k$ vertices on $P$ which we can use to hold as many as $k$ tokens that need to slide into $b$ (in $G'$). In other words, whenever more than one token slides into $b$ in $G'$, we simulate this by sliding the tokens in $P$ onto the $2k$ vertices of $P$ that are free. ◀

As immediate consequences, the following properties hold in an instance where reduction rules R1 to R5 cannot be applied.

▶ **Corollary 2.2.** *Every planetary component must contain at least one token and therefore $G$ can have at most $k$ planetary components, when $k \geq 2$.*

▶ **Corollary 2.3.** *Let $(G, k, I_s, I_t)$ be an instance of* Galactic Token Sliding *where reduction rules R1, R3, and R5 (adjacent black holes rule, absorption rule, and the path reduction rule) have been exhaustively applied. Then, the graph $G$ has diameter at most $O(k^2)$. Moreover, any planetary component has diameter at most $O(k^2)$.*

We now show how the galactic reconfiguration framework combined with the previous reduction rules immediately implies that Token Sliding is fixed-parameter tractable for parameter $k + \Delta(G)$, where $\Delta(G)$ denotes the maximum degree of $G$. Theorem 2.4 immediately implies positive results for graphs of bounded bandwidth/bucketwidth.

▶ **Theorem 2.4.** *Token Sliding is fixed-parameter tractable when parameterized by $k+\Delta(G)$. Moreover, the problem admits a bikernel*[3] *with $k\Delta(G)^{O(k^2)} + (2k + 2k\Delta(G))\Delta(G)$ vertices.*

**Proof.** Let $(G, k, I_s, I_t)$ be an instance of Token Sliding. We first transform it to an instance of Galactic Token Sliding where all vertices are planetary vertices. We then apply all of the reduction rules R1 to R5 exhaustively. By a slight abuse of notation we let $(G, k, I_s, I_t)$ denote the irreducible instance of Galactic Token Sliding.

The total number of planetary components in $G$ is at most $k$ by Corollary 2.2 and the diameter of each such component is at most $O(k^2)$ by Corollary 2.3. Hence the total number of planet vertices is at most $k\Delta(G)^{O(k^2)}$.

To bound the total number of black holes, it suffices to note that no black hole can have a neighbor in $B \cup (A \setminus N[I_s \cup I_t])$. In other words, no black hole can be adjacent to another black hole (since the adjacent black holes reduction rule would apply) and no black hole can be adjacent to a planet without neighboring tokens (otherwise the absorption reduction rule would apply). Hence, combined with the fact that each black hole must have degree at least one, the total number of black holes is at most $(2k + 2k\Delta(G))\Delta(G)$. ◀

## 3 The multi-component reduction rule (R6)

**General idea.** The goal of this section is to show how we can reduce a graph when we have a small vertex separator with many components attached to it. We let $X$ be a subset of vertices and $H$ be an induced subgraph of $G - X$ (for simplicity we assume $G$ is a non-galactic graph in this section). Let $I_s$ and $I_t$ be two independent sets which are disjoint from $H$ and consider a reconfiguration sequence from $I_s$ to $I_t$ in $G$. Let $v$ be a vertex of $H$ and assume that there is a token $t$ that is *projected* on $v$ at some point of the reconfiguration sequence, meaning that the token $t$ is moved from a vertex of $X$ to $v$. This token may stay a few steps on $v$, move to some other vertex $w$ of $H$, and so on until it eventually goes back to $X$. Let this sequence of vertices (allowing duplicate consecutive vertices) be denoted by $v_1 = v, v_2, \ldots, v_r$. We call this sequence the *journey* of $v$ (formal definitions are given in the next subsection).

[3] A kernel where the resulting instance is not an instance of the same problem.

Assume now that the number of connected components attached to $X$ is arbitrarily large. Our goal is to show that one of those components can be safely deleted, that is, without compromising the existence of a reconfiguration sequence if one exists. Suppose that we decide to delete the component $H$. The transformation from $I_s$ to $I_t$ does not exist anymore since, in the reconfiguration sequence, the token $t$ was projected on $v \in V(H)$. But we can ask the following question: Is it possible to simulate the journey of $v$ in another connected component of $G - X$? In fact, if we are able to find a vertex $w$ in a connected component $H' \neq H$ of $G - X$ and a sequence $w_1 = w, \ldots, w_r$ of vertices such that $w_i w_{i+1}$ is an edge for every $i$ and such that $N(w_i) \cap X = N(v_i) \cap X$, then we could project the token $t$ on $w$ instead of $v$ and perform this journey instead of the original journey[4] of $t$. One possible issue is that the number $r$ of (distinct) vertices in the journey can be arbitrarily large, and thus the existence of $w$ and $H'$ is not guaranteed *a priori*. This raises more questions: What is important in the sequence $v = v_1, \ldots, v_r$? Why do we go from $v_1$ to $v_r$? Why many steps in the journey if $r$ is large? The answers are not necessarily unique. We distinguish two cases.

First, suppose that in the reconfiguration sequence, the token $t$ was projected from $X$ to $v$, performed the journey without having to "wait" at any step (so no duplicate consecutive vertices in the journey), and then was moved to a vertex $x' \in X$. Then, the journey only needs to "avoid" the neighbors of the vertices in $X$ that contain a token. Let us denote by $s_1$ the step where the token $t$ is projected on $v$ and by $s_2$ the last step of the journey (that is, the step where $t$ is one move/slide away from $X$). Let $Y$ be the vertices of $X$ that contain a token between the steps $s_1$ and $s_2$. The journey of $t$ can then be summarized as follows: a vertex whose neighborhood in $X$ is equal to $N(v) \cap X$, a walk whose vertices all belong to $H$ and are only adjacent to subsets of $X \setminus Y$, and then a vertex whose neighborhood in $X$ is equal to $N(v_r) \cap X$. In particular, if we can find, in another connected component of $G - X$, a vertex $w$ for which such a journey (with respect to the neighborhood in $X$) also exists, then the we can project $t$ on $w$ instead of $v$. Clearly, the obtained reconfiguration sequence would also be feasible (assuming again no other tokens in the component of $w$).

However, we might not be able to go "directly" from $v_1 = v$ to $v_r$. Indeed, at some point in the sequence, there might be a vertex $v_{i_1}$ which is adjacent to a token in $X$. This token will eventually move (since the initial journey with $t$ in $H$ is valid), which will then allow the token $t$ to go further on the journey. But then again, either we can reach the final vertex $v_r$ or the token $t$ will have to wait on another vertex $v_{i_2}$ for some token on $X$ to move, and so on (until the end of the journey). We say that there are *conflicts* during the journey[5].

So we can now "compress" the path as a path from $v_1$ to $v_{i_1}$, then from $v_{i_1}$ to $v_{i_2}$ (together with the neighborhood in $X$ of these paths), as we explained above. However, we cannot yet claim that we have reduced the instance sufficiently since the number of conflicts is not known to be bounded (by a function of $k$ and/or the size of $X$). The main result of this section consists in proving that, if we consider a transformation from $I_s$ to $I_t$ that minimizes the number of moves "related" to $X$, then (almost) all the journeys have a "controllable" amount of (so-called important) conflicts. Actually, we prove that, in most of the connected components $H$ of $G - X$, we can assume that we have a "controllable" number of important conflicts for every journey on $H$ in a transformation that minimizes the number of token modifications involving $X$. The idea consists in proving that, if there are too many important conflicts during a journey of a token $t$, we could mimic the journey of $t$ on another component to reduce the number of token slides involving $X$. Finally, we will only have to prove that if all the vertices have a controllable number of conflicts (and there are too many components), then we can safely delete a connected component of $G - X$.

[4] We assume for simplicity in this outline that the component of $w$ does not contain tokens.

[5] Actually, there might exist another type of conflict we do not explain in this outline for simplicity.

**Journeys and conflicts.** We denote a reconfiguration sequence from $I_s$ to $I_t$ by $\mathcal{R} = \langle I_0, I_1, \ldots, I_{\ell-1}, I_\ell \rangle$. Let $X \subseteq V(G)$ and $H$ be a component in $G - X$ such that $I_s \cap V(H) = I_t \cap V(H) = \emptyset$. All along this section, we are assuming that tokens have labels just so we can keep track of them. For every token $t$, let $v_i(t)$, $0 \leq i \leq \ell$, denote the vertex on which token $t$ is at position $i$ in the reconfiguration sequence $\mathcal{R}$.

Whenever a token enters $H$ and leaves it, we say that the token makes a journey in $H$. Let $I_i$ denote the first independent set in $\mathcal{R}$ where $v_i(t) \in V(H)$ and let $I_j$, $i \leq j$, denote the first independent set after $I_i$ where $v_{j+1}(t) \notin V(H)$. Then the *journey $J$ of $t$ in $H$* is the sequence $(v_i(t), \ldots, v_j(t))$. The journey is a sequence of vertices (with multiplicity) from $H$ such that consecutive vertices are either the same or connected by an edge. We associate each journey $J$ with a walk $W$ in $H$. The *walk $W$ of $t$* in $H$ is the journey of $t$ where duplicate consecutive vertices have been removed.

We say that a token is *waiting at step $i$* if $v_i(t) = v_{i-1}(t)$; otherwise the token is *active*. Given a journey $J$ and its associated walk $W$, we say that $w \in W$ is a *waiting vertex* if there is a step where the vertex $w$ is a waiting vertex in the journey. Otherwise $w$ is an *active vertex* (with respect to the reconfiguration sequence). So we can now decompose the walk $W$ into waiting vertices and transition walks. That is, assuming the walk starts at $y$ and ends at $z$, we can write $W = yP_0w_0P_1w_1 \ldots w_\ell P_\ell z$, where each $w_i$ is a waiting vertex and each $P_i$ is a *transition walk* (consisting of the walk of active vertices between two consecutive waiting vertices). Note that the transition walks could be empty.

We are interested in why a token $t$ might be waiting at some vertex $w$. In fact, we will only care about waiting vertices that we will call important waiting vertices. Let $w_1, \ldots, w_\ell$ be the waiting vertices of the journey and, for every $i \leq \ell$, let us denote by $[s_i, s'_i]$ the time interval of the reconfiguration sequence where the token $t$ is staying on the vertex $w_i$. Note that $s_i < s'_i$ and when $t$ is active the other tokens are not moving; thus the position of any token different from $t$ is the same all along the interval $[s'_i + 1, s_{i+1}]$ for every $i \leq \ell - 1$.

Let $i < j \leq \ell$ and let $w_i$ be a waiting vertex. We say that $w_j$ is the *important waiting vertex after* $w_i$ if $j > i$ and $j$ is the largest integer such that no vertex along the walk of token $t$ between $w_i$ (included) and $w_j$ (included) is adjacent to a token $t' \neq t$ or contains a token $t' \neq t$ between steps $s'_i$ and $s_j$ (note that the important waiting vertex after $w_i$ might be the last vertex of the sequence). Since token $t$ is active from $s'_i + 1$ to $s_{i+1}$ and is moving from $w_i$ to $w_{i+1}$ during that interval, the important waiting vertex after $w_i$ is well-defined and is at least $w_{i+1}$. Let $Q_{i,j}$ denote the walk in $H$ that the token $t$ follows to go from $w_i$ to $w_j$ (both $w_i$ and $w_j$ are included in $Q_{i,j}$). In other words, $Q_{i,j} = w_iP_{i+1}w_{i+1} \ldots P_jw_j$. Now, note that since $w_j$ is the important waiting vertex after $w_i$ (i.e. we cannot replace $w_j$ by $w_{j+1}$), then we claim that the following holds:

▷ Claim 3.1. If $w_j$ is not the last vertex of the walk $W$, either

- **(i)** there is a token on or adjacent to a vertex of $P_{j+1}w_{j+1}$ (the transition walk after $w_j$) at some step in $[s'_i, s'_j]$ or,
- **(ii)** there is a token on or adjacent to a vertex of $Q_{i,j} - w_j$ in the interval $[s_j, s_{j+1}]$.

We now define the notion of conflicts. Since we cannot replace $w_j$ by $w_{j+1}$, it means that, by definition, there is at least one step $s_q$ in $[s'_i, s_{j+1}]$ where a token $t_q \neq t$ is adjacent to (or on a vertex) $v_q$ of $Q_{i,j+1}$. We call such a step a *conflict*. We say that $(s_q, v_q, t_q)$ is the *conflict triplet* associated to the conflict (we will mostly refer to a triplet as a conflict).

The conflicts of type (i) are called *right conflicts* and the conflicts of type (ii) are called *left conflicts*. It might be possible that $w_j$ is the important waiting vertex because we have (several) left and right conflicts. We say that $w_j$ is a *left important vertex* if there is at least one left conflict and a *right important vertex* otherwise.

If $w_j$ is a left important vertex, we let the *important conflict* $(s_\star, v_\star, t_\star)$ denote the first conflict associated with $Q_{i,j}$ between steps $s_j$ and $s'_j$, i.e., there exists no $s$ such that $s_j \leq s < s_\star \leq s'_j$ such that there is a conflict at step $s$ with a token $t' \neq t$ which is either on $Q_{i,j}$ or incident to $Q_{i,j}$. Note that $v_\star$ cannot be a vertex of $Q_{i,j}$ since that would imply at least one more conflict before $s_\star$, hence $v_\star \in N(V(Q_{i,j}))$. If $w_j$ is a right important vertex, we let $(s_\star, v_\star, t_\star)$ denote the *important conflict* associated with $P_{j+1}w_{j+1}$ between steps $s'_i$ and $s'_j$ as the last conflict associated to $P_{j+1}w_{j+1}$, i.e., there exists no $s$ such that $s'_i \leq s_\star < s \leq s'_j$ and there is a conflict $(s, v_s, t_s)$ such that $v_s$ in $P_{j+1}w_{j+1}$ or incident to $P_{j+1}w_{j+1}$. Note that $v_\star$ cannot be a vertex of $P_{j+1}w_{j+1}$ since that would imply at least one more conflict after $s_\star$, hence $v_\star \in N(V(P_{j+1}w_{j+1}))$. We use $\mathcal{C}(Q_{i,j+1})[s'_i, s'_j]$ to denote all conflict triplets (left and right conflicts) associated with $Q_{i,j+1}$ between steps $s'_i$ and $s'_j$.

To conclude this section, let us remark that the conflicts might be due to vertices of $H$ or vertices of $X$. In other words, for a conflict triple $(s, v, t) \in \mathcal{C}(Q_{i,j+1})[s'_i, s'_j]$, $v$ is an $H$-conflict or an $X$-conflict depending on whether $v$ is in $H$ or in $X$. In what follows we will only be interested in $X$-conflicts. The *$X$-important waiting vertex after* $w_i$ is $w_j$ where $j > i$ is the smallest integer such that $\mathcal{C}(Q_{i,j+1})[s'_i, s'_j]$ contains at least one triplet $(s, v, t')$ where $t' \neq t$, $v \in X$, and $s'_i \leq s \leq s'_j$. Now given a journey we can define the sequence of $X$-important waiting vertices as the sequence $w'_1, \ldots, w'_r$ starting with vertex $w_1$ and such that $w'_{j+1}$ is the $X$-important waiting vertex after $w_j$. What will be important in the rest of the section is the length $r$ of this sequence. If this sequence is short (bounded by $f(k)$), then we can check if we can simulate a similar journey in other components efficiently. If the sequence is long, we will see that it implies that we can find a "better" transformation.

Since we will mostly be interested in how a journey interacts with $X$, we introduce the notion of the $X$-walk associated with journey $J$. The *$X$-walk* is written as $W^X = yP_0w_0P_1w_1 \ldots w_\ell P_\ell z$, where each $w$ is an $X$-important waiting vertex and each $P$ is the walk that the token takes (this walk could have non-important waiting vertices) before reaching the next $X$-important waiting vertex. We call each $P$ in an $X$-walk an *$X$-transition walk*.

**Types and signatures.** Let $X$ be a subset of vertices and $H$ be a component of $G - X$. An *$\ell$-type* is defined as a sequence $IY_1W_1Y_2W_2 \ldots Y_\ell W_\ell Y_{\ell+1}F$ such that for every $i$, $W_i$ is a (possibly empty) subset of $X$ and $Y_i$ is a (possibly empty) subset of $X$ or a special value $\perp$ (the meaning of $\perp$ will become clear later on). We call $I$ the *initial value* and $F$ the *final value* and they are both non-empty subsets of vertices of $X$. The 0-type is defined as $IY_0F$ and we allow $I$ to be equal to $F$. We will often represent an $\ell$-type by $(I(Y_iW_i)_{i\leq\ell}Y_{\ell+1}F)$. Note that if $X$ is bounded, then the number of $\ell$-types is bounded. More precisely, we have:

▶ Remark 3.2. The number of $\ell$-types is at most $(2^{|X|} + 1)^{2(\ell+2)}$.

The neighborhood of a set of vertices $S \subseteq V(H)$ in $X$ is called the *$X$-trace* of $S$. A journey $J$ is *compatible* with an $\ell$-type $IY_1W_1Y_2W_2 \ldots Y_\ell W_\ell Y_{\ell+1}F$ if it is possible to partition the $X$-walk $W$ of $J$ into $W^X = yP_0w_0P_1w_1 \ldots P_\ell w_\ell P_{\ell+1}z$ such that:

- the $X$-trace of each vertex $w_i$ is $W_i$,
- for every walk $P_i$ which is not empty, the $X$-trace of $P_i$ is $Y_i$, i.e., $\cup_{x\in P_i} N(x) \cap X = Y_i$ (note that we can have $Y_i = \emptyset$),
- for every empty walk $P_i$, we have $Y_i = \perp$, and
- the $X$-trace of $y$ is $I$ and the $X$-trace of $z$ is $F$.

The *$\ell$-signature* of a vertex $v \in V(H)$ (with respect to $X$) is the set of all $\ell'$-types with $\ell' \leq \ell$ that can be *simulated* by $v$ in $H$. That is, for every $\ell$-type, there exists a walk $W$ starting at $v$ such that $W = vP_0w_0P_1w_1 \ldots P_\ell w_\ell P_{\ell+1}z$ is compatible with the $\ell$-type if and only if the $\ell$-type is in the signature. Two vertices are *$\ell$-equivalent* if their $\ell$-signatures are the same.

▶ **Lemma 3.3.** *One can compute in* $O^*((2^{|X|}+1)^{2(\ell+2)})$ *the* $\ell$*-signature of a vertex* $v$ *in* $H$*.*

**$X$-reduced sequences and equivalent journeys.** Let $J$ be a journey with exactly $r$ $X$-important waiting vertices (in its $X$-walk). Let $w_i$ and $P_i$ be respectively the $i$-th $X$-important waiting vertex and the $i$-th $X$-transition walk. Let $P_{r+1}$ be the final $X$-transition walk. Let us denote by $W_i$ the neighborhoods of $w_i$ in $X$, and by $Y_i$ the neighborhood of $P_i$ in $X$. Let $I$ and $F$ be the neighborhoods of the initial and final vertices of the walk associated with $J$, respectively. The *type* $T$ *of the journey* $J$ is $I(Y_iW_i)_{i\leq r}Y_{r+1}F$.

▶ **Definition 3.4.** *Two journeys are* $X$-equivalent *whenever the following holds:*
- *They have the same number of* $X$*-important waiting vertices;*
- *The initial and final vertex of the* $X$*-walk have the same* $X$*-trace;*
- *For every* $i$*, the* $X$*-trace of the* $i$*th* $X$*-important waiting vertex is the same in both journeys;*
- *For every* $i$*, the* $X$*-trace of the* $i$*th* $X$*-transition walk is the same in both journeys.*

Let $I, J$ be two independent sets and $X$ be a subset of vertices of $G$. A slide of a token *is related to* $X$ if the token is moving from or to a vertex in $X$ (possibly from some other vertex in $X$). We call such a move an $X$*-move.*

▶ **Definition 3.5.** *A transformation* $\mathcal{R}$ *from* $I$ *to* $J$ *is* $X$-reduced *if the number of* $X$*-moves is minimized and, among the transformations that minimize the number of* $X$*-moves,* $\mathcal{R}$ *minimizes the total number of moves.*

**The multi-component reduction.** Let $H$ be a connected component of $G-X$. The $\ell$-signature of $H$ is the union of the $\ell$-signatures of the vertices in $H$. Let $\mathcal{H}$ be a subset of connected components of $G-X$. We say that $H \in \mathcal{H}$ is $\ell$*-dangerous for* $\mathcal{H}$ if there is a $\ell$-type in the $\ell$-signature of $H$ that appears in at most $\ell$ connected components of $\mathcal{H}$. Otherwise we say that $H$ is $\ell$*-safe*. If there are no $\ell$-dangerous components, we say that $\mathcal{H}$ is $\ell$-safe. One can easily prove the following using iterated extractions:

▶ **Lemma 3.6.** *Let* $\ell = 5|X|k$*. If there are more than* $\ell(2^{|X|}+1)^{2(\ell+2)}+2k+1$ *components in* $G-X$*, then there exists a collection of at least* $2k+1$ *components that are* $\ell$*-safe which can be found in* $f(k,|X|)\cdot n^{O(1)}$*-time, for some computable function* $f$*.*

We can now prove the main result of this section:

▶ **Lemma 3.7.** *Let* $I_s, I_t$ *be two independent sets and* $X$ *be a subset of* $V(G)$*. Let* $\mathcal{R}$ *be an* $X$*-reduced transformation from* $I_s$ *to* $I_t$*. Assume that there exists a subset* $\mathcal{H}$ *of at least* $2k+1$ *connected components of* $G-X$ *that is* $(5|X|k)$*-safe. Then, for every* $C \in \mathcal{H}$*, any journey on the component* $C$ *has at most* $5|X|k-1$ $X$*-important waiting vertices in its* $X$*-walk.*

**Sketch of the proof.** Assume for a contradiction that there exists a safe component $C \in \mathcal{H}$ and a journey $J$ of some token $t$ in $C$ that has at least $5|X|k$ $X$-important waiting vertices. Amongst all such journeys, select the one that reaches first its $(5|X|k)$-th $X$-important waiting vertex. Let us denote by $I(Y_iW_i)_{i\leq 5k}Y_{5k+1}$ the type of the journey $J$ that we truncate after $Y_{5k+1}$. And, let us denote by $v(P_iw_i)_{i\leq 5k}P_{5k+1}$ the partition of the walk into $X$-important waiting vertices and $X$-transition walks (we assume the walk starts at vertex $v \in V(C)$).

For each $X$-important waiting vertex $w_i$, let $(q_i, x_i, t_i)$ be the important conflict associated to it. Since there are at most $k$ labels of tokens and $|X|$ vertices in $X$, there exists a vertex $x \in X$ and a token with label $t'$ such that there exists at least 5 waiting vertices such that the important conflict is of the form $(q, x, t')$ for some $q$. In other words, there exists

$P_{i_1}, \ldots, P_{i_5}$ such that for each $P_i$ we have a triplet $(q, x, t')$ (recall that $q$ denotes the step in the reconfiguration sequence). Let us denote by $s_1, \ldots, s_5$ the steps of those conflicts whose token label is $t'$ and whose vertex in $X$ is $x$ and such that these conflicts are important conflict triples in five different $X$-transition walks.

The rest of the proof consists in proving that, rather than making the initial transformation, we can project $t'$ on an appropriately chosen component $C'$ of $\mathcal{H}$ distinct from $C$ and mimic the journey of $t$ between steps $s_1$ and $s_5$. In other words, $t'$ can simply follow a journey on $C'$ of the same type as that of $t$ between $s_1$ and $s_5$ and we can then safely project it back on $x$ at step $s_5$ without creating any $X$-conflicts. This implies that we can strictly reduce the number of $X$-conflicts, a contradiction to the assumption that the transformation is $X$-reduced. We also manage $H$-conflicts by using the minimality of the journey. ◀

▶ **Lemma 3.8.** *Let $I_s, I_t$ be two independent sets and $X$ be a subset of $V(G)$. If $G - X$ contains at least $4k+2$ $(5|X|k)$-safe components, then we can delete one of those components, say $C$, such that there is a transformation from $I_s$ to $I_t$ in $G$ if and only if there is a transformation in $G - V(C)$.*

**Sketch of the proof.** The proof consists in proving that, since all the journeys have at most $5|X|k - 1$ $X$-important waiting vertices by Lemma 3.7, one can replace a journey in a safe component by a journey in another component either by not creating any conflicts or by creating some conflicts on shorter "time periods" which ensures the procedure converges. ◀

▶ **Corollary 3.9.** *Given a cutset $X$, we can assume that $G - X$ has at most $5|X|k(2^{|X|} + 1)^{2(5|X|k+2)} + 4k + 2 = 2^{O(|X|^2 k)}$ connected components. Moreover, when the number of components is larger we can find a component to delete in $f(k, |X|) \cdot n^{O(1)}$-time.*

## 4 Applications

Due to space restriction we only very briefly explain the main steps of the proofs.

**Planar graphs.** First note that, using galactic reconfiguration, we can reduce the diameter of the graph to $O(k^2)$ by Corollary 2.3. The core of the proof then consists in reducing high degree vertices. If the degree is bounded by a function of $k$, then the conclusion will follow from Theorem 2.4. To reduce the degree, we prove that, for every pair $x, y$ of large degree vertices, 1) $V(G) \setminus \{x, y\}$ does not contain too many components by Corollary 3.9 (Rule R6), 2) the graph can be reduced if there are not too many internally vertex-disjoint paths from $x$ to $y$ and, 3) the graph can be reduced if it contains a few other slightly more general structures. All these results together permit to ensure that $G$ does not contain any large degree vertices after applying all the rules, which completes the proof.

**Chordal graphs of bounded clique number.** A *chordal graph* is a graph with no induced cycle of length at least four. Equivalently, it also has a clique-tree (see full version [3] for formal definitions). If a node $v$ of the clique-tree has large degree then $V(G) \setminus B_v$ (where $B_v$ denotes the bag of $v$) has many connected components, and one of them can be removed by Corollary 3.9. So we can assume that the degree of the clique-tree is bounded. Thus, if the graph is large, the clique-tree should contain a long path $P$. If no vertex belongs to a large fraction of the bags of $P$, then the diameter is large and we can reduce the graph by Corollary 2.3. So a vertex belongs to a large fraction of the bags of $P$. We can actually prove that, there exists a sub-path $P'$ of $P$ and $X \subseteq V(G)$ such that $X$ belongs to all the bags of $P'$ and no other vertex appears in a significant fraction of the bags of $P'$. Together with a few other properties, we prove that some vertices in the bags of $P'$ can be deleted without modifying the existence of a reconfiguration sequence from $I_s$ to $I_t$.

## References


1. Akanksha Agrawal, Ravi Kiran Allumalla, and Varun Teja Dhanekula. Refuting FPT algorithms for some parameterized problems under gap-eth. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 2:1–2:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.2.
2. Valentin Bartier, Nicolas Bousquet, Clément Dallard, Kyle Lomer, and Amer E. Mouawad. On girth and the parameterized complexity of token sliding and token jumping. *Algorithmica*, 83(9):2914–2951, 2021. doi:10.1007/s00453-021-00848-1.
3. Valentin Bartier, Nicolas Bousquet, and Amer E. Mouawad. Galactic token sliding. *CoRR*, abs/2204.05549, 2022. doi:10.48550/arXiv.2204.05549.
4. Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, Yota Otachi, and Florian Sikora. Token sliding on split graphs. In *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, pages 13:1–13:17, 2019. doi:10.4230/LIPIcs.STACS.2019.13.
5. Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In G. Goos, J. Hartmanis, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, Timo Lepistö, and Arto Salomaa, editors, *Automata, Languages and Programming*, volume 317, pages 105–118. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988. Series Title: Lecture Notes in Computer Science. doi:10.1007/3-540-19488-6_110.
6. Marthe Bonamy and Nicolas Bousquet. Token sliding on chordal graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2017. doi:10.1007/978-3-319-68705-6_10.
7. Paul S. Bonsma, Marcin Kaminski, and Marcin Wrochna. Reconfiguring independent sets in claw-free graphs. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 86–97, 2014.
8. Nicolas Bousquet, Arnaud Mary, and Aline Parreau. Token Jumping in Minor-Closed Classes. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory*, volume 10472, pages 136–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-662-55751-8_12.
9. Richard C. Brewster, Sean McGuinness, Benjamin Moore, and Jonathan A. Noel. A dichotomy theorem for circular colouring reconfiguration. *Theor. Comput. Sci.*, 639:1–13, 2016.
10. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
11. Erik D. Demaine, Martin L. Demaine, Eli Fox-Epstein, Duc A. Hoang, Takehiro Ito, Hirotaka Ono, Yota Otachi, Ryuhei Uehara, and Takeshi Yamada. Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science*, 600:132–142, October 2015. doi:10.1016/j.tcs.2015.07.037.
12. Eli Fox-Epstein, Duc A. Hoang, Yota Otachi, and Ryuhei Uehara. Sliding token on bipartite permutation graphs. In *Algorithms and Computation - 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 237–247, 2015.
13. Sevag Gharibian and Jamie Sikora. Ground state connectivity of local hamiltonians. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 617–628, 2015. doi:10.1007/978-3-662-47672-7_50.
14. Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding First-Order Properties of Nowhere Dense Graphs. *Journal of the ACM*, 64(3):1–32, June 2017. doi:10.1145/3051095.

15 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005. doi:10.1016/j.tcs.2005.05.008.

16 Takehiro Ito, Marcin Kaminski, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. On the parameterized complexity for token jumping on graphs. In *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, pages 341–351, 2014.

17 Takehiro Ito, Marcin Kamiński, and Hirotaka Ono. Fixed-Parameter Tractability of Token Jumping on Planar Graphs. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation*, volume 8889, pages 208–219. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-319-13075-0_17.

18 Takehiro Ito, Marcin Kamiński, and Hirotaka Ono. Fixed-parameter tractability of token jumping on planar graphs. In *Algorithms and Computation*, Lecture Notes in Computer Science, pages 208–219. Springer International Publishing, 2014.

19 Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. Reconfiguration of vertex covers in a graph. *IEICE Transactions*, 99-D(3):598–606, 2016.

20 Marcin Kamiński, Paul Medvedev, and Martin Milanič. Complexity of independent set reconfigurability problems. *Theoretical Computer Science*, 439:9–15, 2012.

21 Daniel Lokshtanov and Amer E. Mouawad. The complexity of independent set reconfiguration on bipartite graphs. *ACM Trans. Algorithms*, 15(1):7:1–7:19, 2019. doi:10.1145/3280825.

22 Daniel Lokshtanov, Amer E. Mouawad, Fahad Panolan, M.S. Ramanujan, and Saket Saurabh. Reconfiguration on sparse graphs. *Journal of Computer and System Sciences*, 95:122–131, August 2018. doi:10.1016/j.jcss.2018.02.004.

23 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Comput. Geom.*, 49:17–23, 2015. doi:10.1016/j.comgeo.2014.11.001.

24 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

25 Michał Pilipczuk and Sebastian Siebertz. Kernelization and approximation of distance-r independent sets on nowhere dense graphs. *European Journal of Combinatorics*, 94:103309, May 2021. doi:10.1016/j.ejc.2021.103309.

26 Sebastian Siebertz. Reconfiguration on Nowhere Dense Graph Classes. *The Electronic Journal of Combinatorics*, 25(3):P3.24, August 2018. doi:10.37236/7458.

27 J.A. Telle and Y. Villanger. FPT algorithms for domination in sparse graphs and beyond. *Theoretical Computer Science*, 770:62–68, May 2019. doi:10.1016/j.tcs.2018.10.030.

28 Jan van den Heuvel. The complexity of change. *Surveys in Combinatorics 2013*, 409:127–160, 2013.

29 Marcin Wrochna. Reconfiguration in bounded bandwidth and treedepth. *CoRR*, 2014. arXiv:1405.0847.

30 Marcin Wrochna. Homomorphism reconfiguration via homotopy. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 730–742, 2015.

31 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007. doi:10.4086/toc.2007.v003a006.